Design and Implementation of an Automated Network Monitoring and Reporting Back System

Design and Implementation of an Automated Network Monitoring and Reporting Back System

Rafiullah Khan[a,*], Sarmad Ullah Khan[b]

[a] *Queen's University Belfast, Belfast, United Kingdom, Email: rafiullah.khan@qub.ac.uk*
[b] *Politecnico Di Torino, 10129 Turin, Italy, Email: sarmad.khan@polito.it*

Abstract

The task of network monitoring becomes tedious with increase in the network size, heterogeneity and complexity. The available network monitoring and management solutions are not only expensive but also difficult to use, configure and maintain. Manually pin pointing a faulty device in the large complex network is very tricky and time consuming for network administrators. Thus, an automated system is needed that immediately reports to network administrator about fault type and location as soon as it arises. This paper presents design and implementation of an intelligent network monitoring and reporting back system for large size organizations/industries. It is based on programming open-source tools (such as Nagios and RT) and intelligently integrating them to monitor network devices such as switches and routers. To monitor end-user devices in the network, a software package has been developed using Universal Plug & Play (UPnP) technology. The developed monitoring system immediately notifies network administrator as soon as a network problem arises. The notification message clearly pin points the fault location in network topology, its type and impact on rest of the network. If the network problem is not resolved within the pre-specified deadline, a second notification is sent out to the next responsible person. Thus, all people in the priority list are notified one by one with pre-specified deadlines until the problem is resolved.

Keywords: Network Monitoring, Network Management, Universal Plug & Play, Ticketing System, Reporting Back System, Nagios

1. Introduction

Industry 4.0 revolutionizes industries by making them more connected than ever before. This disruptive innovation connects plethora of heterogeneous devices which are communicating among each other to perform complex tasks [1]. Due to relying mostly on IP based communication, ensuring availability of devices and services is utmost necessary. Thus, an automatic, flexible and scalable network monitoring and management system is essential for industries. The monitoring system constantly checks the whole network topology for any failing component, device or service interruption and immediately notifies the administrator [2]. The notification messages carry information about faulty services and devices which may have been caused due to misconfiguration, overloaded or crashed servers, cyber attack, network cable disconnection or device power disconnection [3, 4]. Due to heterogeneity and complexity in large size industrial networks, manual monitoring the state of each device is very tedious and time consuming task. Further, troubleshooting or pin pointing the fault location and its impact on rest of the network is very challenging [5].

Many network monitoring systems can either monitor core network devices (such as switches and routers) or end-user devices (such as PCs, printers, scanners, etc). Further existing solutions have limited flexibility, scalability and provide insufficient information about the fault location and its impact on rest of the network. This paper addresses network monitoring and management challenges in a more flexible way with the design of a new system. The proposed system is based on programming and integrating open-source tools such as Nagios and Request Tracker (RT) and developing a software package using University Plug & Play (UPnP) technology. The proposed system is not only able to monitor intermediate core network devices but also end-user devices located in private sub-networks. Thus, the network monitoring and management task is divided into two parts: (i) monitoring devices directly accessible from ISP or network control center of the organization, and (ii) monitoring devices located in private LANs.

The proposed network monitoring and management system is applicable to any large size organization or industry. However, this paper focuses on university network as a use case example. Fig. 1 depicts the scenario of a university that has several campuses. Each campus has different departments and each department has several private networks (e.g., classes, laboratories, offices, etc). Thus, a large-size university consists of thousands of network devices. Due to limited IPv4 addressing space, universities are normally allocated certain number of public routable IP addresses. The university ISP or network control center

*Corresponding author

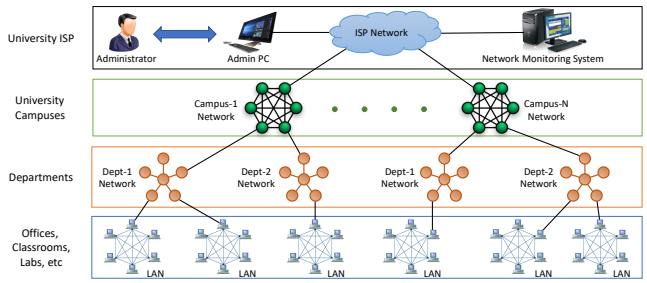


Figure 1: Basic network topology of a large size university.

intelligently utilizes these IP addresses for certain devices e.g., campuses and departments. However, network devices in laboratories, classrooms and offices normally have private IP addresses. In some cases, devices in the private subnets are not directly accessible from the university ISP. Thus, it is necessary that network monitoring and management system should be able to monitor all network devices regardless of their direct accessibility from the university ISP. To this aim, proposed system uses Nagios and RT in the university ISP for monitoring all devices in network topology which are directly accessible. A UPnP based monitoring system is developed for monitoring devices in laboratories, classrooms and offices which are not directly accessible from ISP.

In proposed system, the functionalities of network monitoring are performed by Nagios [6]. Nagios allows easy extension of functionalities and integration of custom modules or plugins. A complete network topology (i.e., consisting of devices directly accessible from university ISP) is created in Nagios and different monitoring services based on ICMP or SNMP are applied on them. Nagios continuously monitors the devices and services running on them and declares them critical or down after making several pre-defined number of attempts. When a network device or service is down, Nagios generates a notification which creates a ticket in RT ticketing system with problematic device information and its impact on rest of the network. RT is responsible for performing the task of fault/problem progress tracking and management [7]. Note that RT is heavily used worldwide by different organizations for tracking and effectively resolving issues/complaints from customers. The scope of RT is quite broad and provides multi-user interface for secure authentication and collaborative management of issues/complaints. The developed UPnP modules are used for monitoring devices in the private LANs. UPnP technology offers auto-discovery, zero-configuration and seamless networking features [8]. Further, UPnP is supported by plethora of heterogeneous devices and does not cause any interoperability and integration issues. The proposed network monitoring system immediately informs network administrator as soon as a network device goes down. If the administrator is busy or could not resolve issue within a set deadline, a new notification is sent out with a new deadline to the next responsible person in the priority list. Thus, all people in the priority list are informed one by one until the problem is resolved. The proposed system is intelligent enough to precisely pin point the problematic device and its impact on rest of the network. Further, a single notification is sent out if a parent device in network topology stops functioning instead of sending multiple notifications for all affected child devices.

The rest of the paper is organized as follows. Section 2 addresses background and related work. Section 3 presents the design of proposed network monitoring and management system. Section 4 provides implementation guidelines. Section 5 presents experimental evaluation of proposed system. Finally, Section 6 concludes the paper.

2. Background and Related Work

Several network monitoring and management tools have been developed over time. The Multi Router Traffic Grapher (MRTG) is one of the most commonly used tool for network monitoring that works on Windows and most Linux distributions [9]. It is written in Perl scripting language and uses Simple Network Management Protocol (SNMP). MRTG generates live visual graphs of all network devices in the HTML pages. These graphs provide information about the traffic load. Manually checking graphs of each network device in MRTG and identifying the problematic/faulty ones is very tedious, inefficient and time consuming task for the network administrator. Another most commonly used networking monitoring tool is Nagios [6]. Nagios is an open-source software tool and is specially designed for monitoring complete infrastructure of enterprise networks for availability of different network services [10]. If a network device is down or any of its network service is down, Nagios raises an alert to the technical staff. Nagios also provides statistics about devices and their services availability.

Several researchers have investigated remote monitoring and management of complex large size networks. Authors in [11] proposed an event monitoring and control system for enterprise networks. The system monitors traffic from network devices to determine their operational status and raises alert to network engineer when failure is detected. Authors in [12] addressed new services in Nagios for monitoring network bandwidth and new form of notifications through email and SMS. These new services increase Nagios scope for even better and improved network monitoring. Authors in [13] addressed how to securely deploy and maintain monitoring services based on Nagios for big enterprise networks. Authors in [2] presented a network monitoring system based on Nagios and designed a more user-friendly and interactive interface. They have extended monitoring functionalities and capabilities through plug-in modules without modifying the Nagios core. Some researchers have also proposed RFID based monitoring and management in industries [14, 15]. Their focus is on

tracking inventory and incoming/outgoing goods e.g., logistic industry. However, presented RFID approaches do not aim to check presence/availability of a network device or service.

Authors in [4] addressed remote monitoring of a LAN that mainly consists of server devices. The system not only monitors availability and operational status of server devices in remote LAN but also their environment. The monitoring system also uses sensor devices and raises SMS alert to administrator in case of a device failure or hazard (e.g., temperature, power fluctuation, humidity, fire, water, etc). Other similar works are addressed in [16, 17, 18]. Normally, organizations have hierarchical network structure for better flexibility and manageability. Further, individuals at network administration have hierarchical nature in terms of responsibilities and privileges. Therefore, authors in [19] proposed efficient authorization function with unified authentication and demonstrated it for hierarchical network monitoring system.

Due to increasing network complexity and heterogeneity of devices, the conventional manual methods for network monitoring and management become tedious as well as time consuming. Even available automated network monitoring options are quite complicated and expensive to deploy and maintain [5, 20, 21, 22]. To ease the installation, configuration and maintenance of network monitoring for large size heterogeneous networks, authors in [23] implemented and testing an ICMP based automated solution that monitors entire network and is also backward compatible to support non-SNMP based devices. Further detailed studies on networking monitoring tools and techniques are available in [24, 25].

Besides reliable and steady operations of the network, authors in [26] also focused on safety and security. They implemented a network monitoring system based on network traffic. A similar work is presented in [27]. Authors in [28] addressed the development of monitoring tool for content-centric networks. Authors in [18] presented a network monitoring and management system for complex university network. However, the monitoring system only monitors network switches and routers and does not monitor network end-user devices. Further, the monitoring system is only applicable to devices directly accessible from ISP and lacks to monitor sub-networks with private IP addresses.

Most previous works were either limited to small LANs or have limited flexibility, limited control and knowledge about fault location. Further, previous network monitoring systems were either limited to core network equipment (e.g., switches, routers) or end-user devices. These are the challenges addressed in this paper with the design of a more efficient, flexible, reliable and scalable network monitoring and management system. The proposed system is based on intelligent integration of Nagios and RT ticketing system [7] along with custom developed software package using UPnP technology.

3. Design of Proposed System

A network monitoring system monitors the whole network topology to ensure that all devices and services are up and running. As soon as a problematic device is detected, it immediately notifies the responsible person. Manually monitoring the state of each device in a complex large network using MRTG graphs or other monitoring tools is very challenging and time consuming. Further, monitoring systems usually lack to monitor devices in low level sub-networks which may not be directly accessible from ISP (e.g., classes, laboratories and office in Fig. 1). The proposed network monitoring system can be used by any organization/industry, but this paper considers a university network as a use case example. The proposed system has been designed with the following objectives:

- Its monitoring functionalities should be continuous and automatic with no or minimal attention required from the network administrator.
- It should offer versatility, portability and dynamic scalability if a device is disconnected or a new device is connected to the network.
- It should be able to monitor core network devices (e.g., switches and routers) as well as end-user devices (e.g., PCs, printers, scanners, copiers, etc).
- It should be able to immediately detect as soon as a network problem arises and notify the network administrator via email or sms.
- It should be vendor independent and capable to monitor plethora of heterogeneous devices from different vendors without interoperability and integration issues.
- It should be able to pin point the exact location of faulty/problematic device in complex network topology and provide enough information about its possible impact on rest of the network.
- It should have low network traffic overhead and should not consume significant resources of the monitored devices.
- It should keep track of changes in the network (e.g., a problem may arise due to change in configurations) and store statistics about a device up and down time.
- It should provide secure remote authentication of the network administrator to the monitoring system.

With these objectives in mind, the design of proposed network monitoring and management system is presented in Fig. 2. It consists of two parts: (i) monitoring of devices directly accessible from university ISP (depicted in Fig. 2(a)), and (ii) monitoring of devices in private LANs e.g., classes, laboratories, offices, etc (depicted in Fig. 2(b)).

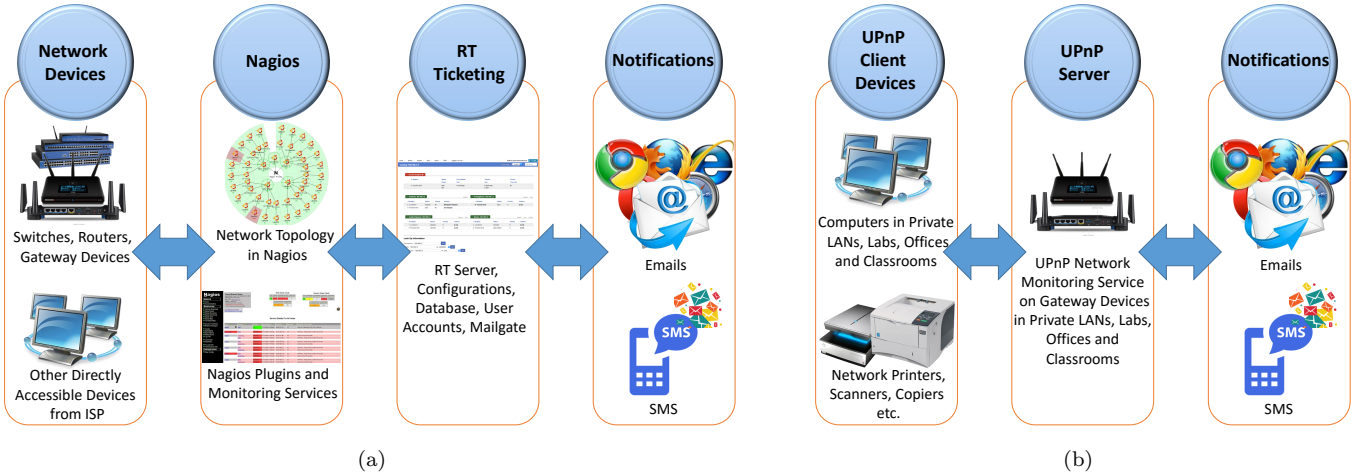


Figure 2: Proposed network monitoring system: (a) Monitoring devices directly accessible from ISP, (b) Monitoring devices in private LANs, labs, classrooms or offices.

An intelligent integration of Nagios and RT ticketing system is used to perform network monitoring and management task for devices directly accessible from university ISP. Whereas, a UPnP based monitoring system is developed for monitoring devices in private LANs.

3.1. Network Monitoring Using Nagios and RT

It can be observed in Fig. 2(a) that main network monitoring and management task is performed by Nagios and RT ticketing system for devices which are directly accessible from the university ISP. Nagios is an open-source tool, specially designed for monitoring status of devices and services. It continuously probes devices and services and sends out notification when any fault is detected [10]. It is widely tested on Linux based systems, however, it is equally suitable for other operating systems as well [6]. It is based on Perl scripting language and a complete network topology can be programmed using specific instructions. While programming the network topology, Nagios also provides options to define a device as child or parent of other devices. When a parent device becomes faulty, all child devices will also become unreachable. However, Nagios can be configured to send only one notification for parent device instead of sending notifications for all affected child devices as well. The parent-child relation is very common in network topologies. An example is shown in Fig. 3 for a university network where Nagios is monitoring different network clusters. Each network cluster can be regarded as a different university campus. Each campus will then have different departments as child devices. If the main router of a campus becomes faulty/unreachable, all its child departments will also become unreachable. For sake of clarity, a Nagios notification about a parent device indirectly indicates that all of its child devices are also affected.

Different types of services can also be defined in Nagios for each device in the network topology. Nagios pro-

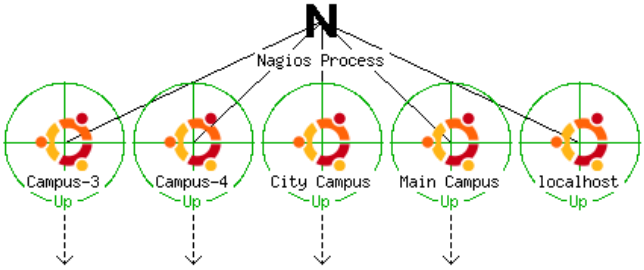


Figure 3: Nagios monitoring different network clusters.

vides web-based interface to graphically analyze the status of devices and services. It also provides statistics about a device/service up and down time and automatically sends notification (e.g., email) when the state of a device changes. The types of services supported by Nagios include but not limited to ICMP PING, SNMP, HTTP, SMTP, NNTP, etc. Nagios monitors devices and services with two terms: (i) status and (ii) type of state. Status represents the present status of a device or service and its possible values are up, down, unreachable and critical. Whereas, the type of state is useful in alert/notification process and can have possible values of soft or hard. The type of state determines the final resulting status of a device or service before sending out notification. To avoid sending false/wrong notifications due to packet loss or glitch in the network, Nagios performs pre-defined number of checks on devices and services before deciding about their final status. The number of pre-defined checks is represented by 'max_check_attempts' when programming and configuring devices and services descriptions. If the number of attempts performed by Nagios are less than max_check_attempts, the non-OK state of a device or service is declared as soft error state. The state is declared as hard error state if the number of checks performed by Nagios exceeds max_check_attempts and each check results

in non-OK state. In hard error state, a device or service is declared as down. Similarly, when a device or service recovers back, Nagios first assigns soft recovery state and then ultimately replaces it with hard recovery state when a device/service consistently appears as OK in number of checks more than max_check_attempts. When the state of a device changes to hard error state, Nagios sends out problem notification. Similarly, when the state of a device changes to hard recovery state, Nagios sends out recovery notification.

The proposed network monitoring and management system also uses RT which is heavily used by different organizations worldwide for offering services to clients and managing their complaints [7]. RT also provides convenient web-based interface with several interesting features (normally required by organizations) such as multiple user accounts, concurrent accessibility and control, email interface, remote accessibility, generation and tracking of notifications/events, etc. RT is open-source and has support for different operating systems including Windows and Linux. Based on the functionalities of RT, a database is required to be installed and properly configured which can be MySQL, ORACLE, POSTGRESQL, etc. The main engine of RT is based on Perl scripting language and requires also Apache web server. Custom scripts can be prepared for RT to send an automatic response to a given event.

In the proposed network monitoring and management system, RT manages the notifications generated by Nagios. Each Nagios notification creates a ticket in RT which is forwarded via email to the network administrator with a specific deadline to resolve the issue. Every RT ticket has a unique identification number and contains information about the problematic device. The responsible person or network administrator can remotely access RT server and work on the ticket. If he is unable to work on the specific issue, he can also assign that ticket to another responsible person. However, if the ticket is not resolved within set deadline (e.g., 3 hours), another notification is sent out to the next responsible person in the priority list and so on (until the issue is resolved).

At present, many organizations use cloud for storage and computation purposes. Migrating an organization's infrastructure as a cloud service is still very limited. To avoid the risk of breaking system and prevent service interruption, many critical industries (e.g., power companies) hesitate migration to the cloud-based orchestration and infrastructure management. However, it is worth mentioning that proposed monitoring system can be used by both, cloud and non-cloud enabled organizations. Nagios has the capabilities to monitor an organization's cloud services or can be deployed itself in the cloud to probe presence of devices and services. Any device or service (physical or virtual) which is accessible over IP network (or accessible from the cloud) can be monitored by Nagios.

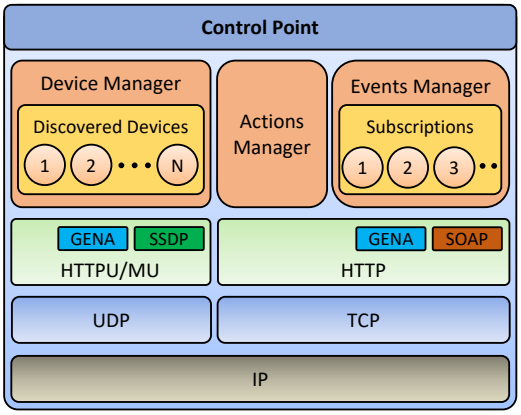


Figure 4: UPnP Control Point.

3.2. Network Monitoring Using UPnP

As shown in Fig. 2(b), a UPnP based service is developed for monitoring devices in the private LANs (e.g., classes, laboratories, offices, etc) which are not directly accessible from university ISP. The network end-user devices are quite heterogeneous and are from different manufacturers. Therefore, a unique monitoring system is required that does not cause any interoperability and integration issues for devices from different manufacturers. The UPnP can be a suitable choice as it is supported by almost every type of network device including routers, PCs, printers, copiers, scanners, etc [8]. Further, UPnP offers autodiscovery, zero-configuration and seamless networking features and does not cause interoperability and integration issues. It enables a device to automatically discover presence of other devices in the network without requiring any configurations. Unlike routers and switches, network end-user devices more frequently connect and disconnect to the network and create uncertainty if a device is down or intentionally powered-down. This makes UPnP even a more better choice that can easily deal with dynamic changes in the network. Further, UPnP is ideal choice from network scalability point of view. As soon as a device is attached to the network, it is automatically discovered by the UPnP server. Thus, network size can dynamically scale in UPnP technology without requiring any configurations from the administrator. In the proposed UPnP based monitoring system, Universally Unique IDentifier (UUID) is used for a device identification purpose. Instead of IP address, the choice of UUID for identification is motivated based on the fact that devices may be using DHCP (i.e., they don't have fixed IP). UUID size is 4 times bigger than IPv4 and can uniquely identify 2^{128} number of devices globally. However, it is worth mentioning that UPnP in proposed system is used only at the boundaries of an organization and monitors only small subnets with limited number of devices (usually less than 100).

In the proposed network monitoring system, a UPnP service is developed and run on the network switch or router that automatically discovers presence of all devices

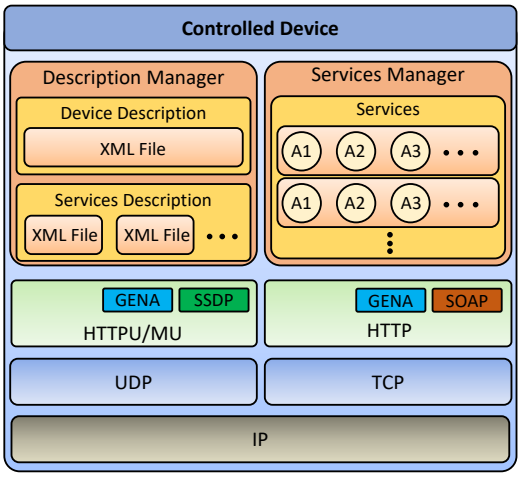


Figure 5: UPnP Controlled Device.

in the network and tracks their power and operational state. The UPnP service automatically sends an email to network administrator as soon as it detects that a network device became unreachable. For automatic discovery of devices presence in the network, UPnP uses Simple Service Discovery Protocol (SSDP). When a network device power state changes, it immediately notifies UPnP service on network switch/router using General Event Notification Architecture (GENA) protocol. Further, a UPnP device also uses Simple Object Access Protocol (SOAP) to send commands (e.g., to notify UPnP service on switch/router that the device is powered-down intentionally by a user). Thus, UPnP service on switch/router precisely knows the reason that why a network device became unreachable. It then informs the network administrator via email and provides information about the unreachable device.

3.2.1. UPnP Device Types

The UPnP device architecture has specified two types of devices: (i) Controlled Device (CD) and (ii) Control Point (CP). The CP functionalities are similar to a client and the CD functionalities are similar to a server. The CP sends requests/commands to the CD which are processed and responded accordingly. Fig. 4 shows the basic building blocks of a UPnP CP. Device Manager handles the list of devices discovered in the LAN and acquires their descriptions. Action Manager has the ability to invoke actions or send requests to the CD. Event Manager handles events based on state variables such as changes in the power state of a discovered device. It also periodically renews subscription of registered devices and deletes a device whose subscription has expired. This subscription renewal and expiry feature is very useful to determine when a network device becomes unreachable.

Fig. 5 shows the basic building blocks of a UPnP CD. The CD offers one or more services to the CP. Description Manager handles device description and also provides it to the CP. The description is expressed in XML format

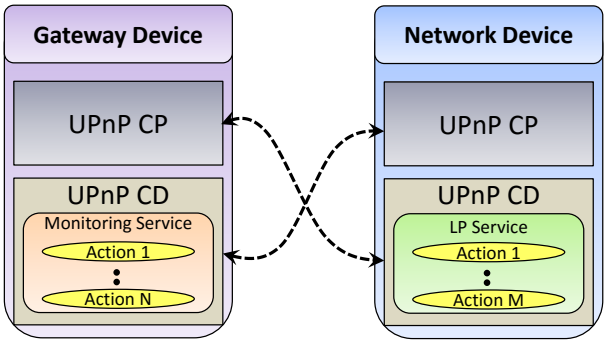


Figure 6: Design of two-way UPnP based command and control framework.

using standard UPnP formatting. Description Manager also periodically advertises CD presence to the CP to renew its subscription lease. Services Manager handles the implementation of different services and actions. The CP can request a service by invoking different actions on the CD. After receiving a request, Services Manager executes or performs a specific task. It also keeps the CP updated through state variables about any change in its operational or power state.

3.2.2. Design of UPnP Service

In general UPnP communication framework, one device implements a CP (e.g., a PC) and other device implements a CD (e.g., a printer). However, in proposed monitoring system, each device implements a CP as well as a CD in order to achieve two way command and control features. The CP on one device enables it to send commands/requests to the CD on other device. The design of UPnP based monitoring system is shown in Fig. 6. The Low-Power (LP) service needs to be executed on all network devices which are being monitored. However, the monitoring service needs to be executed on router/switch or any other LAN device that performs the monitoring tasks. It is worth to mention that LP service also implements a kernel module that tracks changes in the power state of a device and immediately notifies the monitoring service through its CP. When a device receives sleep or shutdown command from user, it takes few seconds to freeze or stop all operations and indeed enter into sleep or shutdown state. This time is more than enough for kernel module to notify the monitoring service. It is also applicable if a network device enters into sleep state due to staying idle for a period of time. The kernel module can also detect if a device battery is low which is quite useful feature for battery powered devices. Based on the information provided by kernel module on a network device, the monitoring service can determine the reasons behind a device unavailability. If a network device subscription expires and becomes unreachable without any notification sent by kernel module, the monitoring service simply assumes that the device network cable or power cable is accidentally disconnected or some misconfiguration took place on the device. In either

case, router/switch (i.e., monitoring service) uses a mail-client and notifies network administrator via email about the fault/problem. The UPnP based monitoring system does not require any network topology to be programmed or configured in the monitoring service. The monitoring service automatically detects the dynamic connection/disconnection of new or existing network devices and their power state transitions.

4. Implementations

This section provides implementation guidelines for the proposed network monitoring and management system. It consists of several open-source tools as well as custom developed modules. Proper installation, configuration and integration of RT, Nagios and custom developed UPnP based software tools are strictly necessary for achieving the desired objectives. This section presents implementation guidelines in generic way and uses university network as a use case example. By following the specified implementation guidelines, a network administrator can develop the proposed monitoring system for his own organization/industry. In short, this section addresses: (i) installation and configuration of RT, (ii) installation and configuration of Nagios, (iii) programming/formation of network topology, (iv) application of monitoring services on network devices, (v) interfacing RT with Nagios, (vi) development of UPnP client software for end-user devices and (vii) development of UPnP server software for gateway devices (e.g., switches/routers).

4.1. Request Tracker Installation & Configurations

The main engine of RT is based on Perl scripting language and requires also Apache web server. A database is also required which can be MySQL, ORACLE or POSTGRESQL, etc. The complexity in RT installation depends on the operating system. It has been observed that RT installation/configuration on Red Hat operating system is very challenging as many dependencies need to be installed and configured manually. However, Ubuntu or Debian operating system provides much ease in installation of dependencies through 'apt-get install' feature. Prior to installation of RT, it is necessary to install and configure Apache web-server, Postfix, Lynx, MySQL server, libdbd-pg-perl (i.e., Perl DBI driver for the PostgreSQL database server) and libapache-dbi-perl (i.e., interface connecting apache server to database via perl's DBI). After successful installation of these dependencies, RT (i.e., request-tracker) installation should complete without any complications.

During the configuration of Postfix, the system requests to specify Fully Qualified Domain Name (FQDN). The FQDN should be specified carefully as it is a key factor allowing remote global access to RT server. Once RT has been installed, the following important configurations (depending on the organization) need to be specified in 'RT_SiteConfig.pm':

```
Set($rtname, 'rtName')
Set($Organization, 'organization')
Set($CorrespondAddress , rt@FQDN)
Set($CommentAddress , rt-comment@FQDN)
Set($WebPath , ''/rt")
Set($WebBaseURL , ''http://FQDN/rt")
Set($DatabaseType, $typemapmysql)
Set($DatabaseUser , 'name')
Set($DatabasePassword , 'password' )
```

The Apache web server must be restarted for configurations to take effect. Finally, the RT instance can be accessed in any web browser at 'http://FQDN/rt'. The default user name is 'root' and password is 'password'. RT allows to create different queues where each queue stores tickets related to specific category/group of events. Fig. 7 depicts queue configuration options in RT. It can be observed in Fig. 7 that a queue 'University Network Monitoring' is created as a use case example. A ticket is generated in this queue every time state of a network device changes. The ticket holds information about the faulty/problematic network device. RT also allows to create different user accounts and necessary rights can be assigned to each account for a specific queue as shown in Fig. 8. Further, multiple users can also be grouped together and rights can be assigned to them on global basis. In proposed network monitoring and management system, pending tasks (about faulty/problematic devices) are represented as tickets in RT. A ticket can be assigned to a specific user account and has different attributes (ticket owner, ticket priority, queue, time worked, time left, watchers, status, etc). The owner and requester are default watchers for a ticket. However, additional watchers can also be defined. Priorities can be assigned to tickets in RT within the range of 0-99 (i.e., 99 as the highest priority). A ticket can be given initial and final priority, which increment or decrement based on time left.

4.2. Nagios Installation & Configurations

Nagios installation is comparatively simpler than RT. Executing following command as root user in Ubuntu/Debian operation system completes the installation of Nagios (latest version 3):

```
# sudo apt-get install nagios3
```

After successful installation, a password is assigned to web user account using following command in terminal:

```
# htpasswd -c /etc/nagios3/htpasswd.users
                                    username
```

The Nagios monitoring system can be accessed in any web browser at 'http://FQDN/nagios3' using default login credentials. Since Nagios has to be interfaced with RT in proposed network monitoring and management system, the RT contact should be defined in 'contacts_nagios.cfg'. The following is definition of RT contact:

```
define contact {
 contact_name                    RT
```

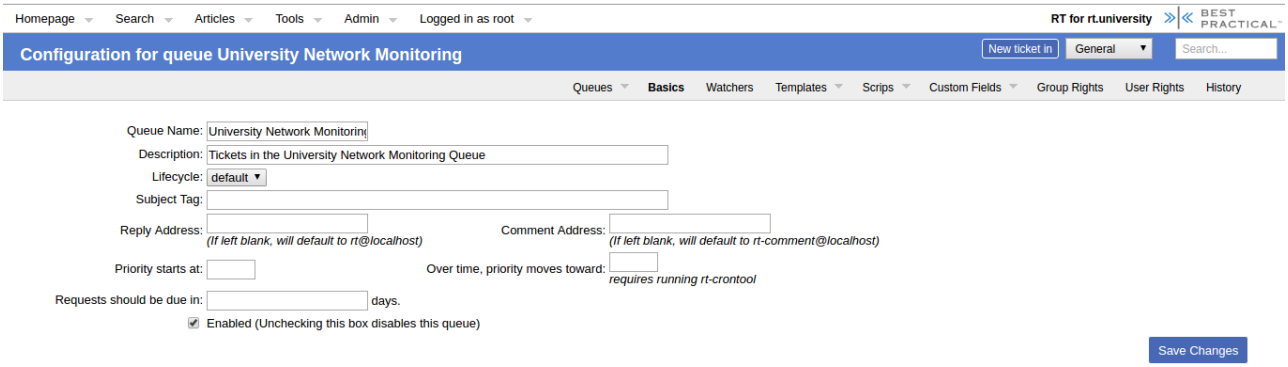


Figure 7: Creating and configuring queue in RT.

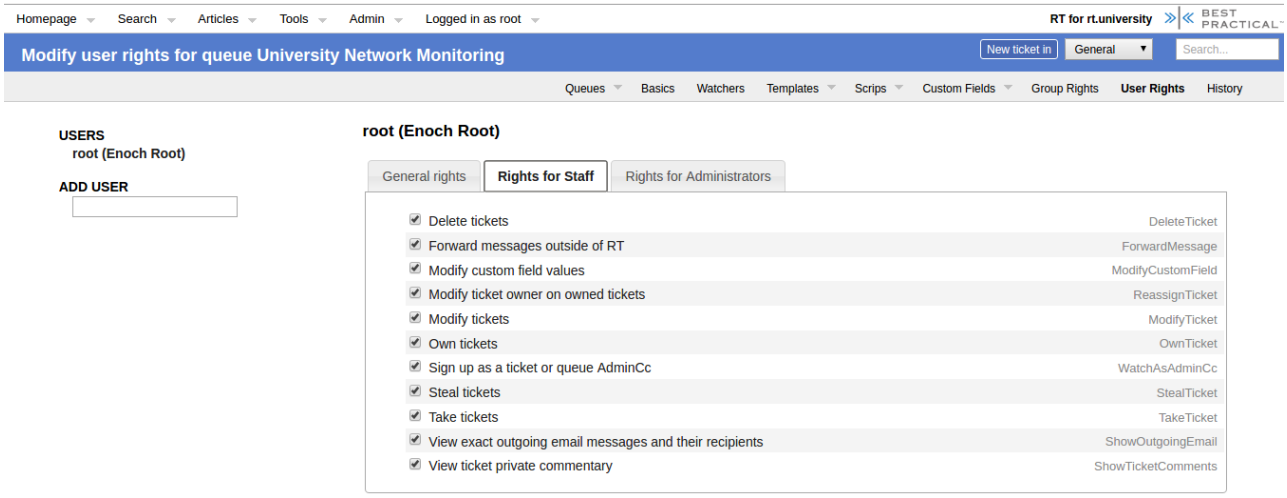


Figure 8: Assigning user rights to the created queue.

```
 alias                              Request-Tracker
 service_notification_period        24x7
 host_notification_period           24x7
 service_notification_options       c,w,r,u
 host_notification_options          r,d
 service_notification_commands      notify-service-by-email
 host_notification_commands         notify-host-by-email
 email                              rt@host.FQDN
}
```

Additional contacts can also be defined based on the requirement. Nagios will inform the contacts when the state of a network device changes. In the above contact definition, Nagios will send notifications to 'rt@host.FQDN' which will generate tickets in RT about problematic/faulty network devices. Once all the contacts are defined, a contact group is created which contains all the contact names.

```
define contactgroup {
 contactgroup_name    Network-admins
 alias                Network and Nagios admins
 members              RT, other-contacts
}
```

Once all the configurations are completed, the Nagios should be restarted for configurations to take effect.

```
# /etc/init.d/nagios3 restart
```

4.3. Formation of Network Topology in Nagios

Since Nagios is performing the network monitoring functionalities, complete network topology needs to be programmed (as shown in Fig. 9(a)). Note, Nagios topology consists of devices which are directly accessible from university ISP. In complex large-size networks, devices have normally parent child relationships. Thus, parent devices should be specified for each device in Nagios. A separate configuration file should be created in '/etc/nagios3/conf.d/' directory for each network device and has the following generalized definition:

```
define host {
 use          generic-host
 host_name    DeviceName
 alias        DeviceAlias
 address      [Device's IP Address]
 parents      [Device's parent if any]
}
```

After defining all of the devices in network topology, a group should be defined in 'hostgroups_nagios2.cfg' that contains all relevant devices as its members. Multiple groups may be defined with specified members based on the type of devices and organization needs. The basic definition of a group is as follows:

```
define hostgroup {
 hostgroup_name      GroupName
 alias               GroupAlias
 members             Device1, Device2,...
}
```

4.4. Defining Monitoring Services for Network Devices

Once network devices and groups are defined, the monitoring services can be applied on them. Based on the organization, Nagios may monitor services such as ICMP PING, HTTP, SSH, etc on devices in defined groups. Nagios monitors not only the devices but also services. It is also possible that some services on a device are up/running while the others are down. The basic definition of ICMP PING service (in 'services_nagios2.cfg') on network devices is as follows:

```
define service {
 hostgroup_name        GroupName, other-groups
 service_description   PING
 check_command         check_ping!100.0,20%!500.0,60%
 use                   generic-service
 notification_interval 0 ; for re-notification, set > 0
}
```

After defining services, restart the Nagios for configurations to take effect.

```
# /etc/init.d/nagios3 restart
```

4.5. Interfacing RT with Nagios

Following the installation and configuration of RT and Nagios, they should be properly integrated or interfaced together. Nagios will perform the network monitoring task whereas RT will be used for management activities. Whenever a network device becomes unreachable or recovers back, Nagios sends out a notification which generates a ticket in the specified queue in RT. The ticket can be assigned to different administrative persons based on their availability to resolve the network issue.

An important component in interfacing RT with Nagios is 'rt-mailgate'. Thus, an alias should be created in 'aliases' using the following:

```
rt: ''|rt-mailgate --queue 'RT Queue Name'
    --action correspond --url http://FQDN/rt"
rt-comment:  ''|rt-mailgate --queue 'RT Queue Name'
    --action comment --url http://FQDN/rt"
```

These configurations will direct Nagios notifications to the specified queue in RT. It is necessary that the name of queue should exactly match the queue created in RT. Further, created users and groups in RT should have appropriate rights to own, modify, delete or forward tickets.

4.6. UPnP Client for Network Devices in Labs and Offices

The UPnP based software tools are developed for monitoring of network devices which are not directly accessible from university/organization ISP e.g., certain subnets in classes, laboratories, offices, etc. The design of basic software package is depicted in Fig. 6 where detailed architectures of UPnP control point and controlled device are shown in Fig. 4 and Fig. 5, respectively. The UPnP client software for network end-user devices was developed in Linux operating system using standard C/C++ programming language. For ease in the implementation, libupnp and boost libraries were used. The libupnp is open-source libraries for implementation of UPnP features which are

compliant with UPnP device architecture v1.0. While the boost libraries make easier the implementation of networking tasks. A kernel module was also developed in C programming language that tracks changes in the operational state of device. The role of kernel module is very important in understanding why a network device became unreachable. The developed kernel module can effectively detects if the device received sleep or shutdown instruction or if its battery is low or if the network or power cable is disconnected. It immediately notifies the UPnP monitoring server on gateway device (i.e., switch/router) which in turn sends email notification to network administrator.

4.7. UPnP Monitoring Server for Gateway Devices in Labs and Offices

The UPnP monitoring server was also developed in Linux operating system using standard C/C++ programming language along with libupnp and boost libraries. It implements monitoring functionalities and a mail client to send email notifications to network administrator. The UPnP enables monitoring server on gateway devices (e.g., switches, routers) to seamlessly and automatically discover network end-user devices and track their operational status through kernel module on them. It is worth to mention that each network device has a subscription expiry time at the monitoring server. If a network device stops functioning due to unknown reasons and could not notify monitoring server, the monitoring server will ultimately know it upon the expiry of its UPnP subscription period. Thus, the whole monitoring process works autonomously and seamlessly without requiring any specific attention from the user.

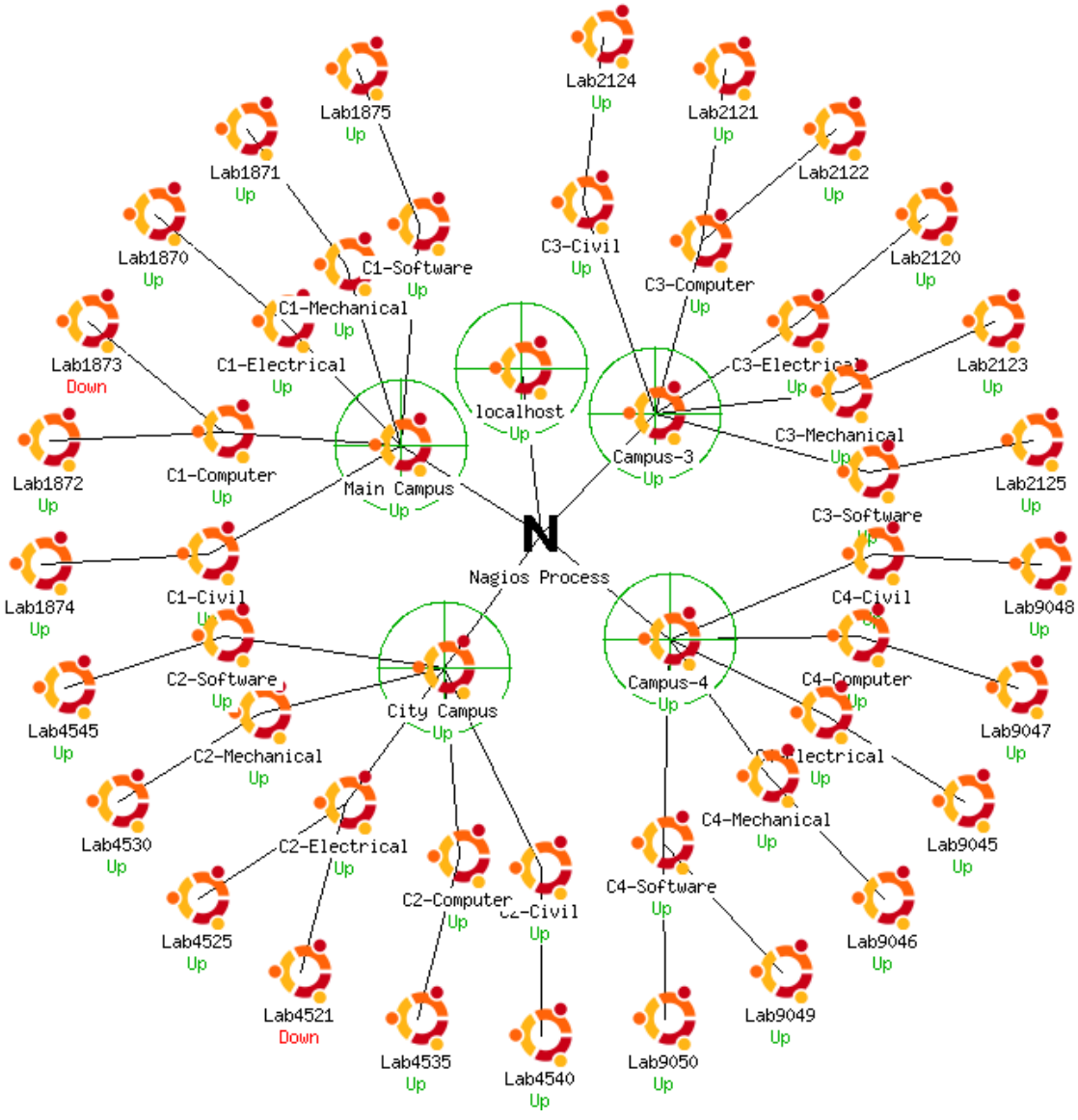
5. Evaluation of Proposed System

The proposed network monitoring and management system is applicable for any large size organization. However, experimental evaluations were performed considering a complex large-size university network. Fig. 9(a) depicts network topology programmed in Nagios. It assumes that all these devices are network switches or routers or other devices which are directly accessible from the university ISP. For experimentation purpose, CISCO switches with the support of SNMP were used. These switches can be manually configured and services (e.g., ICMP PING, SNMP, SSH, etc) can be enabled on them. At present, experiments are based on ICMP PING service which is enabled on all devices in network topology. Fig. 9(a) depicts that university network consists of four campuses where each campus consists of five departments (i.e., civil, computer, electrical, mechanical and software engineering departments). These departments in Fig. 9(a) are entry level routers or switches. Each department can have different private sub-networks which can be classrooms, laboratories and offices. Fig. 9(a) depicts that all switches/routers are up and running except Lab1873 and Lab4521.

Nagios was configured with 10 seconds interval to probe presence of network devices. It declares a device in 'soft error state' when unreachable in the first attempt. Nagios was also configured with 10 total failed attempts before declaring a device in 'hard error state'. To verify the correct functioning, two CISCO switches (i.e., Lab1873 and Lab4521) were made unavailable (either by disconnecting network cable or unplugging switch power supply). Nagios was working correctly and declared both switches 'down' as can be observed in Fig. 9(a). The configured probe interval (i.e., 10 seconds) should be chosen based on the requirement of organization. It is worth to mention that it is directly linked with monitoring system traffic overhead. The repeated attempts (i.e., 10 attempts) before declaring a device in 'hard error state' plays an important rule in preventing false notifications due to packet loss or glitch in the network.

When Nagios declares a device as 'down', it sends out notification that generates a ticket in RT. It can be observed in Fig. 9(b) that tickets were generated for both down switches (i.e., Lab1873 and Lab4521) in 'University Network Monitoring' queue in RT. The network administrator has pre-specified deadline of 1 hour to resolve the issue otherwise another notification is sent out to next responsible person based on priority and so on. It is also possible that a network administrator disowns the ticket if he feels unable to resolve a specific network issue. Such ticket can be owned by other users on RT based on their skills and capabilities. It was also observed that Nagios sends out notification when a faulty/unreachable device recovers back. To this aim, switch Lab2121 in computer engineering department of campus-3 was made unreachable (i.e., disconnected its network cable) and then its connectivity issue was resolved again. It can be observed in Fig. 9(b) that a ticket was also generated in RT based on the resolve notification sent by Nagios. As specified in Section 4.2, additional contacts can also be defined in Nagios who will receive notifications based on specified state changes e.g., a top level responsible person. Fig. 10 depicts email notification sent by Nagios to top level responsible person when switch Lab4521 (i.e., in electrical engineering department in city campus) recovers back. Nagios also provides detailed device/host state information as depicted in Fig. 11. It shows how long a device has been in its present state, is it soft state or hard state, time of last status check, any notifications sent out, etc.

Experiments were also performed for developed UPnP based software tools. For the sake of simplicity and availability of equipment, the UPnP monitoring server was executed on a low-power Raspberry Pi v2 and located inside the local network. However, the monitoring server software is portable enough and can be easily cross-compiled for embedded processors of network switches and routers. The UPnP zero-configuration, auto-discovery and seamless networking features were also verified. As soon as a new device is attached to the network, the UPnP monitoring server immediately detects it and tracks its operational



(a)

10 highest priority tickets I own — Edit

#	Subject	Priority	Queue	Status
5	Network Device (ID: 4521) is Down. This network device is located in Electrical engineering department in city campus	0	University Network Monitoring	open
6	Network Device (ID: 2121) is up again	0	University Network Monitoring	new

10 newest unowned tickets — Edit

#	Subject	Queue	Status	Created	
4	Network Device (ID: 1873) is Down. This network device is located in computer engineering department in main campus	University Network Monitoring	new	12 minutes ago	Take

My reminders

Quick search — Edit

Queue	new	open	stalled
General	-	-	-
University Network Monitoring	2	1	-

Dashboards — Edit

(b)

Figure 9: Interaction between Nagios and RT: (a) Map of the network monitored by Nagios, (b) Tickets generated in RT when state of any network device changes.

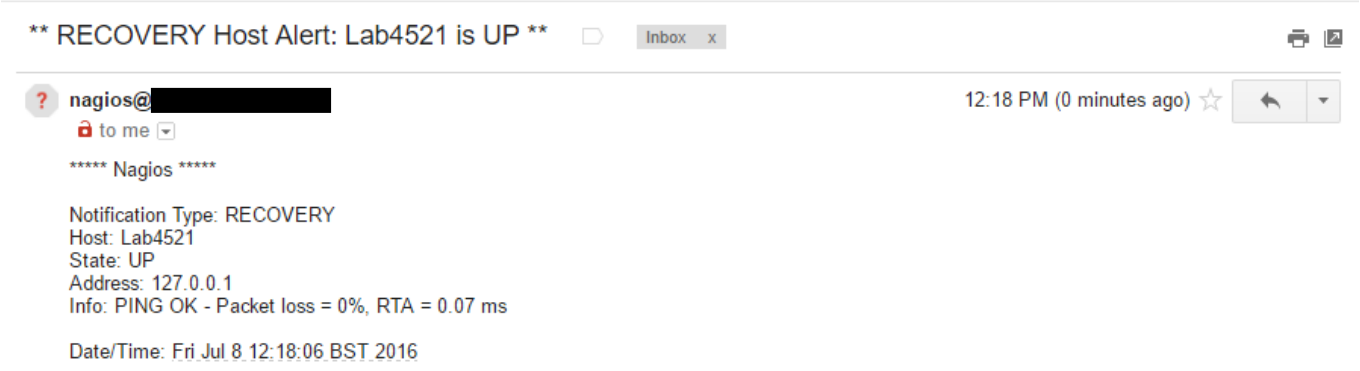


Figure 10: Email notification by Nagios when a network device is recovered and up again.

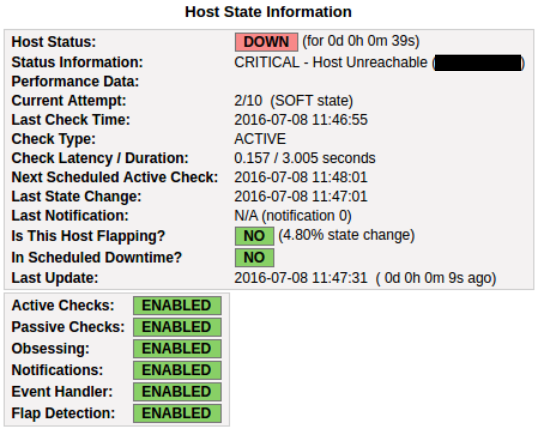


Figure 11: Host state information in Nagios.

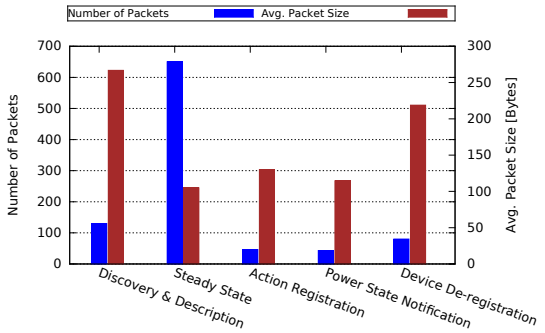


Figure 12: Overhead analysis in terms of average packet size and number of packets exchanged during different UPnP events.

status with the help of kernel module. All operations are autonomous without requiring any user attention or input. Further, the developed UPnP based tools are very flexible and do not cause any interoperability or integration issues between devices from different manufacturers. It was observed that the UPnP based monitoring system is very scalable and also adopts to dynamic changes in the network (e.g., new devices added or existing devices disconnected from the network).

To test the functionalities of UPnP monitoring server, a network device i.e., a PC was put into standby state. The kernel module in UPnP client software immediately tracked it and notified to the UPnP monitoring server. Based on the configurations, the UPnP monitoring server sent out email notification to network administrator informing about the network problem. Similarly, a resolve email notification was sent out when the network device recovered back and a message was received from its kernel module. The experimental tests were also successfully performed by disconnecting the network cable of a device. In this case, the UPnP monitoring server performed decisions based on subscription expiry (i.e., device is unreachable) and renewal (i.e., device has recovered and renewed its subscription). The UPnP based monitoring system provides more flexibility as it is supported by plethora of heterogeneous devices including PCs, printers, scanners, copiers, routers, switches, etc.

The communication overhead is normally considered most critical factor for network monitoring and management systems. High overhead may consume available bandwidth and impair routine network operations. Thus, communication overhead was measured for the proposed UPnP based network monitoring system. The experiment lasted 12 minutes during which UPnP client on a network device registered with UPnP monitoring server, registered an action, notified its power/operational state, stayed in idle state and finally de-registered with UPnP monitoring server. Fig. 12 depicts traffic overhead considering average packet size and total number of packets exchanged during different UPnP events. It was observed that most packets were very small in size (i.e., periodic presence advertisements and subscription renewals during steady state). The infrequent packets which were exchanged during discovery/registration and de-registration phases were big in size due to carrying complete device description in XML format. Fig. 13 provides more clarity by depicting the total number of Bytes exchanged during different UPnP events.

The communication overhead was also analyzed in terms of real-information inside each packet, additional overhead

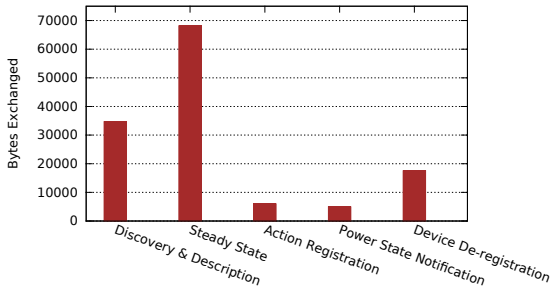


Figure 13: Overhead analysis in terms of total Bytes exchanged during different UPnP events.

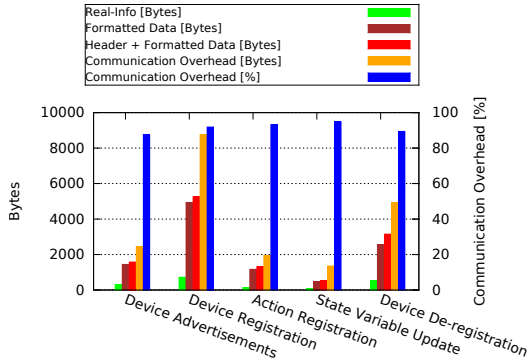


Figure 14: Overhead analysis in terms of percentage of real information and additional overhead Bytes inside packets during different UPnP events.

due to XML formatting, additional overhead due to packet headers and total overhead due to UPnP semantics. It can be observed in Fig. 14 that device advertisement, registration and de-registration generate significant overhead due to exchange of multiple packets and download of complete device and service descriptions. However, such packets are very infrequent. During steady state, packets are very small in size and have less overhead Bytes. It can be observed that the UPnP overhead is not significantly high to impair routine network operations. This confirms the suitability of proposed UPnP based monitoring system for local network devices.

The proposed monitoring system is expected to be a part of the organization's network. Like existing devices, the additional devices and servers used by the monitoring system will also be protected by the organization's existing firewall, VPN or other security system. However, most organizations are vulnerable to insider attacks e.g., a PC compromised by a malware installed through spear phishing email. The presence probe messages in proposed monitoring system do not carry any critical device specific information. Therefore, eavesdropping on such messages does not raise any major privacy concern. Through eavesdropping, the attacker may identify that a monitoring system exists that probes presence of devices and services. If such packets (e.g., ICMP PING, SNMP, etc) are dropped by the attacker, the proposed monitoring system can easily detect it as the specific device or service will become unreachable. This will trigger email notifications from the monitoring server alerting network administrator about the issue. Note that such insider attack scenarios are very rare in practice as organizations/industries normally have effective security system detecting and preventing cyber attacks. Thus, monitoring system does not need a separate security mechanism but instead operates under the protection of an organization's existing security system.

6. Conclusions & Future Work

Network monitoring and management is always a challenging task for large organizations with complex network topology. It is tedious for network engineers/administrators to manually check the operational state of hundreds or potentially thousands of devices in the network topology. Even, a network fault for few hours can cause significant financial loss to the organization along with the loss of customers satisfaction. Many organizations still have complaint system where an employee/customer/client registers a complaint about network connectivity before the network administrators start troubleshooting. Network fault reporting, identification and fixing could become much faster with the design of an automatic network monitoring and management system.

This paper presented an automated network monitoring and management system that eases the responsibilities of network administrators with the goal of keeping servers and devices operational 24/7. The proposed system is intelligent enough to quickly identify faulty/problematic device, its location and impact on rest of the network. It immediately notifies the network administrator via email as soon as a network problem arises. The proposed system offers portability, dynamic network scalability and does not cause interoperability and integration issues between plethora of heterogeneous devices from different manufacturers. Further, all the monitoring functionalities are continuous and automatic without requiring any specific input or attention from the administrator. Further, the proposed system provides flexibility for network administrators. If the administrator is busy and does not resolve network fault/problem within pre-specified deadline, a new email notification is sent to the next responsible person in the priority list and so on. The unique features and their experimental validation prove the significance of proposed network monitoring and management system.

This paper addressed the proposed system for a large university network as a use case example. However, it is equally applicable for other organizations. In current implementations, the UPnP based monitoring software is not interfaced/integrated with RT. Future work will focus on further improvement where each notification from UPnP based monitoring system will generate a ticket in RT. This

will further increase the scope of proposed network monitoring and management system.

References

[1] Y. Lu, Industry 4.0: A Survey on Technologies, Applications and Open Research Issues, in: Journal of Industrial Information Integration, Vol. 6, 2017, pp. 1–10.

[2] C. Issariyapat, P. Pongpaibool, S. Mongkolluksame, K. Meesublak, Using Nagios as a Groundwork for Developing a Better Network Monitoring System, in: 2012 Proceedings of PICMET '12: Technology Management for Emerging Technologies, 2012, pp. 2771–2777.

[3] A. G. Finogeev, A. A. Finogeev, Information Attacks and Security in Wireless Sensor Networks of Industrial SCADA Systems, in: Journal of Industrial Information Integration, Vol. 5, 2017, pp. 6–16.

[4] A. Kijazi, M. Kisangiri, Multifunctional Network Monitoring System using SMS, in: Science, Computing and Telecommunications (PACT), 2014 Pan African Conference on, 2014, pp. 143–147. doi:10.1109/SCAT.2014.7055149.

[5] A. Mahajan, H. Joshi, S. Khajuria, A. Verma, ICMP, SNMP: Collaborative Approach to Network Discovery and Monitoring, in: International Journal of Smart Sensors and Ad Hoc Networks, Vol. 1, 2012, pp. 8–12.

[6] Nagios - The Industry Standard In IT Infrastructure Monitoring, in: https://www.nagios.org.

[7] D. R. R. S. D. Chamberlain, R. Foley, J. Vincent, Rt essentials, in: Installation, Chap:2, pp.9-18, 2005.

[8] Universal Plug & Play (UPnP) - The Open Connectivity, in: https://openconnectivity.org.

[9] MRTG - The Multi Router Traffic Grapher, in: https://www.mrtg.com.

[10] M. Schubert, A. Hay, D. Bennett, et. al, Nagios3 Enterprise Network Monitoring, in: Designing Configurations for Large Organizations, Chap:2, pp.25-84, 2008.

[11] M. Al-Mukhtar, S. Khalil, A System for an Enterprise Remote Network Monitoring and Fault Management based on SMS and WAP, in: Wireless Communications and Applications (ICWCA 2012), IET International Conference on, 2012, pp. 1–5. doi:10.1049/cp.2012.2102.

[12] M. A. A. bin Mohd Shuhaimi, Z. binti Zainal Abidin, I. binti Roslan, S. binti Anawar, The New Services in Nagios: Network Bandwidth Utility, Email Notification and SMS Alert in Improving the Network Performance, in: Information Assurance and Security (IAS), 2011 7th International Conference on, 2011, pp. 86–91. doi:10.1109/ISIAS.2011.6122800.

[13] A. Rogozhkin, Deploying Nagios Monitoring Services on Secured Red Hat Enterprise Linux 3 Environment, in: Global Information Assurance Certification Paper - SANS Institute, 2005.

[14] C. Zhai, Z. Zou, Q. Chen, L. Xu, L.-R. Zheng, H. Tenhunen, Delay-Aware and Reliability-Aware Contention-free MFTDMA Protocol for Automated RFID Monitoring in Industrial IoT, in: Journal of Industrial Information Integration, Vol. 3, 2016, pp. 8–19.

[15] S. Alyahya, Q. Wang, N. Bennett, Application and Integration of an RFID-enabled Warehousing Management System A Feasibility Study, in: Journal of Industrial Information Integration, Vol. 4, 2016, pp. 15–25.

[16] M. Bhamare, T. Malshikare, R. Salunke, P. Waghmare, GSM based LAN Monitoring and Controlling, in: International Journal of Modem Engineering Research, Vol. 2, 2012, pp. 387–389.

[17] R. Ciprian, B. Lehman, Modeling Effects of Relative Humidity, Moisture, and Extreme Environmental Conditions on Power Electronic Performance, in: 2009 IEEE Energy Conversion Congress and Exposition, 2009, pp. 1052–1059. doi:10.1109/ECCE.2009.5316423.

[18] R. Khan, S. U. Khan, R. Zaheer, M. I. Babar, An Efficient Network Monitoring and Management System, in: International Journal of Information and Electronics Engineering, Vol. 3, 2013, pp. 122–126.

[19] K. Matsuura, Y. Seki, M. Sano, T. Ueta, Design and Implementation of Organizational Authorization for a Network Monitoring System, in: 2014 Second International Symposium on Computing and Networking, 2014, pp. 605–607. doi:10.1109/CANDAR.2014.70.

[20] U. H. Rao, Challenges of Implementing Network Management Solution, in: International Journal of Distributed and Parallel Systems, Vol. 2(5), 2011, pp. 67–76.

[21] R. Voicu, I. C. Legrand, C. Dobre, A Monitoring Framework for Large Scale Networks, in: Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on, 2011, pp. 429–432. doi:10.1109/ICCP.2011.6047909.

[22] X. Chen, Y. Mao, Z. M. Mao, J. V. der Merwe, DECOR: DEClarative network management and OpeRation, in: ACM SIGCOMM Computer Communication Review, Vol. 40, 2010, pp. 61–66. doi:10.1145/1672308.1672321.

[23] A. Sinha, L. Sejwal, N. Kumar, A. Yadav, Implementation of ICMP based Network Management System for Heterogeneous Networks, in: Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on, 2015, pp. 382–387.

[24] A. Cecil, A Summary of Network Traffic Monitoring and Analysis Techniques, in: Washington University in St. Louis, Project Report.

[25] C. So-In, A Survey of Network Traffic Monitoring and Analysis Tools, in: Washington University in St. Louis, Project Report.

[26] X. Li, T. Jiang, Design and Implementation of the Campus Network Monitoring System, in: Electronics, Computer and Applications, 2014 IEEE Workshop on, 2014, pp. 117–119. doi:10.1109/IWECA.2014.6845571.

[27] X. Wang, L. Wang, B. Yu, G. Dong, Studies on Network Management System Framework of Campus Network, in: Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on, Vol. 2, 2010, pp. 285–289. doi:10.1109/CAR.2010.5456547.

[28] W. Kang, B. Sim, J. Kim, E. Paik, Y. Lee, A Network Monitoring Tool for CCN, in: World Telecommunications Congress (WTC), 2012, 2012, pp. 1–3.